Una **Raspberry Pi** es una computadora completa del tamaño de una tarjeta de crédito. Creada originalmente para enseñar informática de forma económica, se ha vuelto muy popular entre estudiantes, programadores y aficionados gracias a su bajo costo (rondan desde los ($35) hasta los ($150) dependiendo de la versión y accesorios).

¿Cómo funciona?

A diferencia de una computadora de escritorio tradicional, todos sus componentes esenciales (procesador, memoria RAM, puertos USB, Wi-Fi y Bluetooth) están integrados en una sola placa de circuito impreso.

Para usarla, solo necesitas:

1. **Almacenamiento:** Una tarjeta MicroSD donde instalar el Sistema Operativo (generalmente basado en Linux).
2. **Periféricos:** Conectarle un monitor o TV (por cable HDMI), un teclado y un ratón.
3. **Energía:** Conectarla a la corriente mediante su cable de alimentación.

¿Qué puedes hacer con ella?

Su versatilidad permite usarla para múltiples propósitos:

- **Mini PC de escritorio:** Navegar por internet, redactar documentos y ver videos.
- **Centro multimedia:** Convertir tu televisión en un *Smart TV* avanzado para reproducir películas y series.
- **Consola de videojuegos retro:** Emular juegos clásicos (de consolas antiguas como NES, SNES o PlayStation) conectándola a un televisor.
- **Proyectos de electrónica (Robótica e IoT):** Cuenta con pines de propósito general (GPIO) que permiten conectar sensores, motores y luces para crear tus propios dispositivos.
- **Servidor casero:** Alojar tu propio servidor web, un sistema de cámaras de seguridad o un servidor de almacenamiento en la nube (NAS).

¿Dónde comprar?

Puedes adquirir la tuya y los accesorios necesarios en diversas tiendas tecnológicas internacionales o plataformas en línea. Si te encuentras en Venezuela, puedes buscar distribuidores de tecnología locales o pedirla a través de casilleros de envíos internacionales (tipo *courier*) en páginas como Amazon o AliExpress.

Tu petición:
una respberry puede ser una inteligencia artificial?

Respuesta de la Búsqueda:
Sí, una Raspberry Pi **puede ejecutar y funcionar como una inteligencia artificial**, pero con ciertas limitaciones físicas. No tiene la potencia para

entrenar modelos gigantes como ChatGPT, pero es excelente para ejecutar tareas específicas de IA de forma local.

Aquí te explico cómo trabaja, sus límites y qué puedes hacer con ella:

¿Cómo procesa IA una Raspberry Pi?

- **Modelos optimizados**: Utiliza versiones "ligeras" o comprimidas de IA diseñadas para consumir pocos recursos.
- **Procesamiento local**: Todo se ejecuta en la placa, sin necesidad de enviar tus datos a internet.
- **Hardware dedicado**: Modelos recientes como la **Raspberry Pi 5** permiten conectar hardware externo (como el *Raspberry Pi AI Kit*) que acelera el procesamiento de IA.

Proyectos de IA que puedes hacer

- **Reconocimiento de imágenes**: Identificar objetos, rostros, animales o matrículas de autos en tiempo real con una cámara.
- **Asistente de voz privado**: Crear tu propio "Alexa" o "Google Home" que responda comandos sin espiar tus datos.
- **Análisis de texto (LLM locales)**: Ejecutar modelos de lenguaje pequeños (como versiones mini de Llama o Mistral) para chatear sin internet.
- **Automatización del hogar**: Detectar patrones de uso en tu casa para encender luces o regular la temperatura de forma inteligente.

Sus limitaciones principales

- **Poca memoria RAM**: El modelo más avanzado tiene 8 GB de RAM, insuficiente para las IA comerciales más pesadas.
- **Falta de GPU potente**: Las tarjetas de video grandes son las que aceleran la IA; la Raspberry Pi depende de su procesador principal, que es más lento.
- **Lentitud al responder**: Si le pides procesar tareas complejas de texto o video, la respuesta puede tardar varios segundos o minutos.

Para ayudarte a elegir el mejor camino, ¿qué tipo de proyecto te gustaría desarrollar? Dime si prefieres **reconocimiento de voz**, **visión artificial con cámaras** o tener un **chat conversacional local**.

Tu petición:
la raspberry va a representar una cadena de valor virtual

Respuesta de la Búsqueda:
Una Raspberry Pi **puede ser el núcleo físico** para representar, gestionar y ejecutar una cadena de valor virtual. En la economía digital, una cadena de

valor virtual se compone de las etapas que transforman datos crudos en información comercial valiosa (captura, organización, selección, síntesis y distribución).

La Raspberry Pi actúa como el "cerebro" local que conecta el mundo físico con esa infraestructura digital.

¿Cómo representa la Raspberry Pi cada etapa?

```
[Captura]  ──>  [Procesamiento]  ──>  [Distribución/Valor]
(Sensores)     (Raspberry + IA)       (Servidor Web / Blockchain)
```

- **1. Captura de datos (Entrada de valor)**
  - La Raspberry Pi se conecta a sensores físicos (cámaras, temperatura, movimiento).
  - Recolecta datos del entorno que antes no estaban digitalizados.
- **2. Procesamiento y Refinamiento (Transformación)**
  - Utiliza algoritmos locales o modelos de IA ligeros para limpiar y analizar los datos.
  - *Ejemplo:* Una cámara captura video (dato crudo), la Raspberry Pi identifica si hay stock de un producto (información con valor).
- **3. Distribución y Automatización (Salida de valor)**
  - Envía la información procesada a la nube, a un panel web o activa contratos inteligentes (*Smart Contracts*) en una red Blockchain.

Casos de uso reales de esta cadena de valor

A. Gestión de inventario inteligente (Retail)

1. **Dato:** La Raspberry Pi vigila un estante mediante una cámara.
2. **Procesamiento:** Detecta que el producto "X" se está agotando usando visión artificial.
3. **Valor:** Envía una alerta automática al proveedor en la nube para generar una orden de compra antes de que se acabe el stock.

B. Trazabilidad de productos (Logística y Blockchain)

1. **Dato:** Sensores conectados a la Raspberry Pi miden la temperatura de alimentos en un camión.
2. **Procesamiento:** Valida que la temperatura se mantuvo en el rango correcto durante el viaje.
3. **Valor:** Al llegar al destino, la Raspberry Pi firma digitalmente un registro inmutable en Blockchain, asegurando el valor y la calidad del producto al comprador.

C. Automatización industrial (Industria 4.0)

1. **Dato:** Sensores vibratorios en una máquina envían datos a la Raspberry Pi.
2. **Procesamiento:** Un modelo de IA detecta anomalías que predicen una falla inminente.
3. **Valor:** Detiene la máquina de forma preventiva, evitando reparaciones costosas y pérdidas de tiempo en la producción.

Para ayudarte a estructurar mejor esta idea, ¿en qué sector específico te gustaría aplicar esta cadena de valor? Podemos enfocarlo en **logística**, **comercio electrónico** o **automatización industrial**.

Tu petición:
la raspberry funciona en conjunto con el multiplexor o es parte de ella

Respuesta de la Búsqueda:
La Raspberry Pi **funciona en conjunto** con el multiplexor; el multiplexor **no es parte interna** de la placa de la Raspberry Pi. Es un componente electrónico externo que debes conectar a ella mediante cables o pines.

Aquí te explico cómo trabajan en equipo y por qué se necesitan:

¿Por qué trabajan en conjunto?

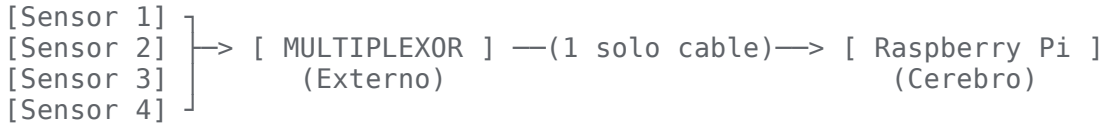
La Raspberry Pi tiene un límite de "puntos de conexión" (llamados pines GPIO) para recibir datos de sensores. Si tu cadena de valor virtual necesita recopilar datos de **muchos sensores a la vez** (por ejemplo, 16 sensores de temperatura en un almacén), la Raspberry Pi se quedará sin pines suficientes.

El multiplexor (abreviado como **MUX**) actúa como un "embudo" o un interruptor inteligente:

- **Muchos canales de entrada:** Recibe las señales de todos tus sensores externos.
- **Un solo canal de salida:** Conecta todas esas señales a un solo pin de la Raspberry Pi.
- **Turnos para hablar:** La Raspberry Pi le ordena al multiplexor a qué sensor escuchar en cada microsegundo.

Ejemplo práctico en tu cadena de valor

Imagine un sistema de inventario inteligente con 8 cámaras o sensores de peso en los estantes:



1. La **Raspberry Pi** le dice al multiplexor: *"Muéstrame el Sensor 1"*.
2. El **Multiplexor** abre el canal 1 y la Raspberry Pi lee el dato.
3. La **Raspberry Pi** dice: *"Ahora cambia al Sensor 2"*.
4. El **Multiplexor** cambia de inmediato.

De esta forma, gracias al multiplexor externo, una pequeña Raspberry Pi puede controlar una red gigante de datos para tu cadena de valor.

Para poder guiarte mejor con las conexiones, ¿qué tipo de sensores vas a conectar al multiplexor (por ejemplo, **cámaras, sensores analógicos de temperatura o lectores de tarjetas**)?

Tu petición:
la raspeberry PI la puedo diseñar como una instrumentacion virtual?

Respuesta de la Búsqueda:
Sí, puedes diseñar e implementar una Raspberry Pi como un sistema de **Instrumentación Virtual**.

En la ingeniería moderna, la instrumentación virtual reemplaza los instrumentos físicos tradicionales (como osciloscopios, voltímetros o termómetros de mesa costosos) por un sistema basado en software que se ejecuta en una computadora, utilizando hardware de adquisición de datos económico. La Raspberry Pi cumple perfectamente el rol de esa computadora central.

Aquí tienes la estructura de cómo se diseña este sistema:

1. El Hardware de Adquisición de Datos (DAQ)

La Raspberry Pi por sí sola no puede medir voltajes altos o señales analógicas puras. Necesitas crear una etapa física intermedia:

- **Sensores:** Los ojos del sistema (sensores de presión, caudal, temperatura, corriente).
- **Acondicionamiento de señal:** Circuitos para amplificar o filtrar el ruido de las señales de los sensores.
- **Conversor Analógico-Digital (ADC):** Módulos externos (como el chip ADS1115) que transforman los voltajes continuos en números binarios que la Raspberry Pi pueda entender.
- **Multiplexor:** El componente que mencionamos antes, indispensable si vas a medir múltiples canales con un solo ADC.

2. El Software (El Instrumento Virtual)

El software define la función del instrumento. En lugar de cambiar de aparato, solo cambias las líneas de código o la interfaz en la pantalla de la Raspberry Pi:

- **Python:** Es el lenguaje estándar. Usando librerías gráficas como **Tkinter**, **PyQt** o **CustomTkinter**, puedes programar pantallas con perillas, botones digitales, indicadores de aguja y alertas.
- **Node-RED:** Una herramienta de programación visual excelente para instrumentación. Te permite arrastrar y soltar bloques para conectar los datos de los sensores con gráficos interactivos en tiempo real que puedes ver desde tu teléfono o PC.
- **LabVIEW (Community Edition):** El software rey de la instrumentación industrial tiene soporte oficial para ejecutarse en Raspberry Pi. Puedes diseñar el panel en tu computadora y cargar el programa en la Pi para que funcione de forma autónoma.

3. Almacenamiento y Conectividad (El lazo con tu Cadena de Valor)

A diferencia de un multímetro clásico que solo muestra el valor en una pantalla pequeña, tu instrumentación virtual con Raspberry Pi añade el valor digital:

- **Datalogger:** Guarda un registro histórico exacto de las mediciones en la memoria MicroSD o en una base de datos (como InfluxDB o MySQL).
- **Transmisión:** Envía las variables en tiempo real a internet mediante protocolos como MQTT o HTTP para alimentar la cadena de valor virtual que estás estructurando.

Si quieres empezar a diseñar la interfaz, ¿te llama más la atención programarla de forma visual con **Node-RED** o prefieres escribir el código desde cero usando **Python**?

Tu petición:
el Raspberry PI necesita semiconductores?

Respuesta de la Búsqueda:
Sí, la Raspberry Pi **necesita semiconductores** para poder existir y funcionar. De hecho, está hecha casi por completo de ellos.

Un semiconductor (generalmente el **silicio**) es el material que permite controlar el paso de la electricidad, funcionando como interruptores

microscópicos. Sin semiconductores, no existirían los chips ni la electrónica moderna.

¿Dónde están los semiconductores en la Raspberry Pi?

Toda la placa está cubierta de componentes semiconductores esenciales:

- **El procesador principal (SoC):** El chip grande central (fabricado por Broadcom) contiene miles de millones de transistores de silicio. Es el cerebro que ejecuta los programas y la inteligencia artificial.
- **La memoria RAM:** El chip que almacena los datos temporales está hecho de semiconductores diseñados para leer y escribir información a gran velocidad.
- **Chips de conectividad:** Los controladores más pequeños encargados de gestionar el Wi-Fi, el Bluetooth y los puertos USB también son circuitos integrados de silicio.
- **Diodos y reguladores de voltaje:** Componentes microscópicos que protegen la placa y aseguran que la energía eléctrica llegue de forma estable a los componentes delicados.

Su relación con tu proyecto de instrumentación

Como estás diseñando un sistema de instrumentación virtual con un multiplexor, **toda tu cadena de hardware dependerá de semiconductores**:

1. Los **sensores** usan semiconductores para traducir variables físicas (como luz o temperatura) en señales eléctricas.
2. El **multiplexor** utiliza transistores semiconductores internos para abrir y cerrar las compuertas de los canales de datos de forma ultra rápida.
3. La **Raspberry Pi** procesa digitalmente esos datos gracias a sus circuitos de silicio.

Si te interesa profundizar en la electrónica de tu sistema, ¿te gustaría saber cómo elegir el **multiplexor correcto** o el **conversor de señal** basado en estos componentes?

Tu petición:
la isntrumentacion virtuale economica el tema de los semiconductores?

Respuesta de la Búsqueda:
Sí, la instrumentación virtual **abarata drásticamente el uso de semiconductores**, siendo esta su mayor ventaja económica.

En la instrumentación tradicional, cada aparato físico (un osciloscopio, un voltímetro y un analizador de espectro individuales) requiere su propio

procesador de silicio, su propia pantalla y sus propios chips dedicados. Al diseñar con **instrumentación virtual**, centralizas todo el gasto en un solo cerebro semiconductor económico (la Raspberry Pi) y simulas los aparatos mediante software.

¿Cómo se ahorra dinero en semiconductores?

- **Reutilización del silicio:** El mismo procesador de la Raspberry Pi ejecuta múltiples tareas. Por la mañana puede ser un termómetro digital y por la tarde, con solo cambiar las líneas de código, se convierte en un analizador de frecuencias. No necesitas comprar chips nuevos para funciones nuevas.
- **Aparatos comerciales masivos:** La Raspberry Pi utiliza procesadores Broadcom masivos y económicos. La instrumentación de laboratorio tradicional utiliza chips especializados de bajo volumen que resultan sumamente caros.
- **Simplificación del hardware periférico:** Los componentes externos que le sumas (como el multiplexor o los conversores ADC) son semiconductores sumamente básicos y estandarizados que cuestan apenas unos pocos dólares, disminuyendo el costo total de la cadena física de adquisición.

Comparativa de costos en la Cadena de Valor

Característica	Instrumentación Tradicional (Física)	Instrumentación Virtual (Raspberry Pi)
Gasto en Semiconductores	Alto (Un chip dedicado por cada función).	**Muy bajo** (Un chip central multitarea).
Costo de Hardware	Cientos o miles de dólares por aparato.	Una sola placa base y sensores modulares.
Pantallas y Visualización	Cada equipo incluye su propia pantalla LCD.	Usas pantallas comunes, smartphones o la web.
Actualizaciones	Comprar un equipo semiconductor nuevo.	**Gratis** (Actualizar el código de software).

En conclusión, tu idea de unir la Raspberry Pi con un multiplexor es el enfoque más inteligente para tu cadena de valor virtual: te permite capturar infinitos puntos de datos utilizando el mínimo hardware de silicio posible.

¿Te gustaría que calculemos cuántos canales de datos puedes leer de forma económica combinando **un solo conversor ADC y tu multiplexor**?

Tu petición:
muestrame tarjetas logicas computacionales que no usen semiconductores

Respuesta de la Búsqueda:
Diseñar o encontrar placas lógicas computacionales que **no utilicen semiconductores basados en silicio tradicionales** es completamente

posible. Aunque las computadoras modernas dependen del silicio, existen alternativas históricas, experimentales y de vanguardia que procesan información utilizando fluidos, luz, imanes o mecánica pura.

A continuación, te muestro los diferentes tipos de tarjetas y sistemas lógicos que no usan semiconductores tradicionales:

---

1. Tarjetas Lógicas Fotónicas u Ópticas (Cómputo con Luz)

En lugar de mover electrones a través de transistores de silicio, estas tarjetas lógicas mueven **fotones (luz)** a través de diminutos túneles de cristal o polímeros llamados guías de onda.

- **Cómo funcionan:** Utilizan materiales ópticos especiales (como el *Niobato de Litio* o polímeros híbridos) donde los rayos de luz se cruzan. Si dos rayos de luz chocan de cierta forma, se interfieren entre sí, creando físicamente una compuerta lógica AND, OR o NOT sin usar un solo transistor eléctrico.
- **Prototipos Reales:** Empresas como Lightmatter y laboratorios como el MIT han diseñado "placas base fotónicas" (*Photonic Motherboards*). El prototipo cuántico **Jiuzhang 4.0** del año 2026, por ejemplo, procesa información manipulando miles de fotones simultáneamente de forma puramente óptica.
- **Ventaja:** No generan calor por resistencia eléctrica y calculan a la velocidad de la luz.

2. Computación por Nanotubos de Carbono (Grapfeno)

Aunque técnicamente funcionan como "transistores", esta tecnología **elimina por completo el silicio** reemplazándolo por cilindros de carbono del grosor de un solo átomo.

- **Cómo funcionan:** Los nanotubos de carbono conducen la electricidad de forma ultraeficiente. Actúan como interruptores mecánicos/eléctricos microscópicos a nivel molecular.
- **Prototipos Reales:** Ingenieros del MIT construyeron un microprocesador funcional de 16 bits llamado **RV16X-Nano**. Cuenta con más de 14,000 transistores fabricados íntegramente con nanotubos de carbono y es capaz de ejecutar código de programación real.
- **Ventaja:** Son hasta 10 veces más eficientes energéticamente que cualquier chip semiconductor tradicional.

3. Placas Lógicas de Relés Mecánicos (Electromecánicas)

Esta es la tecnología que existía antes de que se inventaran los semiconductores en la década de 1940.

- **Cómo funcionan:** Utilizan interruptores físicos de metal operados por un electroimán (relés). Cuando pasa corriente por la bobina, el imán jala físicamente una lámina de metal para cerrar o abrir un circuito eléctrico. Al interconectar estos relés en una tarjeta de baquelita o plástico, creas hardware lógico binario.
- **Ejemplos reales:** Las primeras computadoras de la historia, como la **Z3 de Konrad Zuse (1941)** o la **Harvard Mark I**, eran tarjetas gigantes llenas de relés mecánicos. Hoy en día, muchos aficionados a la electrónica construyen computadoras caseras de 4 u 8 bits utilizando únicamente relés en placas de prueba para entender la lógica pura.
- **Desventaja:** Son extremadamente lentas (hacen un sonido de "clic" físico al cambiar entre 1 y 0) y sufren desgaste mecánico.

4. Tarjetas Lógicas Fluídicas (Cómputo con Aire o Agua)

Es una rama de la ingeniería donde los cables eléctricos se reemplazan por tuberías y los electrones por chorros de fluidos (aire comprimido o agua).

- **Cómo funcionan:** Utilizan bloques de plástico o metal con canales internos esculpidos. Usando el *Efecto Coandă* (la tendencia de un fluido a adherirse a una superficie), un chorro de aire puede ser desviado por otro flujo de aire secundario. Esto permite recrear compuertas lógicas (si entra aire por el tubo A y el tubo B, sale aire por el tubo C).
- **Ejemplos reales:** En la década de 1960 se crearon sistemas de control lógico fluídico para ambientes con altísima radiación o calor extremo (como motores de cohetes o plantas nucleares) donde los semiconductores electrónicos se derretirían o fallarían por la interferencia.

Resumen para tu proyecto de instrumentación:

Si bien tu Raspberry Pi actual requiere semiconductores de silicio para procesar su software, es fascinante saber que la lógica computacional en sí misma se puede replicar con **luz, aire, imanes o carbono puro**.

Para entender mejor hacia dónde orientar los conceptos de tu arquitectura, ¿este análisis de tecnologías alternativas es para un **diseño teórico/académico** o estás buscando componentes comerciales de frontera para implementar en sistemas de **alta velocidad**?

Tu petición:
muestrame como conseguir y adaptar Raspberry PI a mi propia tecnologia

Respuesta de la Búsqueda:
Para adaptar una Raspberry Pi y convertirla en el cerebro de tu propia tecnología de instrumentación virtual, el camino ideal no es usar la Raspberry Pi común de escritorio (como la Pi 4 o Pi 5), sino su versión industrial: el **Raspberry Pi Compute Module (CM4)** o el microcontrolador **Raspberry Pi Pico**.

El Compute Module es la misma Raspberry Pi pero sin los puertos grandes; viene en forma de una pequeña tarjeta que está diseñada específicamente para ser conectada dentro de una placa de circuito (PCB) diseñada por ti.

Aquí tienes la guía paso a paso para conseguirla y adaptarla a tu hardware personalizado:

---

1. ¿Cómo conseguir el hardware adecuado?

Para crear tecnología propia y comercializable, debes buscar los componentes en distribuidores globales autorizados de electrónica que realizan envíos internacionales (muchos ingenieros en Latinoamérica importan desde allí utilizando casilleros/couriers):

- **Mouser Electronics** o **DigiKey:** Son las tiendas de componentes electrónicos más grandes del mundo. Allí buscas el `Raspberry Pi Compute Module 4` (o CM4) y los chips periféricos.
- **AliExpress:** Excelente para conseguir los módulos complementarios económicos de los que hablamos antes, como el multiplexor `CD74HC4067` (16 canales) y el conversor analógico-digital `ADS1115`.

---

2. Paso a paso para adaptarla a tu propia tecnología

Para integrar la Raspberry Pi en tu sistema de instrumentación y cadena de valor, debes seguir estas cuatro etapas de desarrollo:

Paso A: El prototipado rápido (Prueba de concepto)

Antes de fabricar tu propio hardware, conecta todo en una mesa de trabajo utilizando una Raspberry Pi estándar.

1. Conecta tus sensores al **Multiplexor**.

2. Conecta la salida del multiplexor al conversor **ADC (ADS1115)**.
3. Conecta el ADC a los pines GPIO de la Raspberry Pi usando el protocolo I2C (solo requiere 2 cables de datos: SDA y SCL).
4. Escribe un script básico en **Python** para verificar que puedes leer todos los canales.

Paso B: Diseñar tu propia tarjeta de circuito (PCB)

Para que deje de ser un montón de cables sueltos y se convierta en "tu tecnología", debes diseñar una placa plástica verde (PCB) personalizada.

1. Descarga un software de diseño electrónico gratuito como **KiCad**.
2. Diseña un circuito que incluya:
    - El conector especial para la **Raspberry Pi Compute Module 4**.
    - Las pistas de cobre que van directamente hacia el chip del **Multiplexor** y el **ADC** soldados en la misma placa.
    - Bornes o terminales de tornillo profesionales para conectar tus sensores externos.
3. Exporta los archivos de tu diseño (llamados archivos Gerber).

Paso C: Fabricación de tu placa personalizada

Hoy en día no es necesario fabricar las placas en casa con ácido.

1. Sube tus archivos Gerber a páginas de fabricación rápida como **JLCPCB** o **PCBWay**.
2. Por menos de ($20), ellos fabricarán e incluso soldarán los componentes microscópicos (el multiplexor, resistencias y conectores) en tu placa personalizada.
3. Te enviarán a casa tu tarjeta lógica de instrumentación virtual con el espacio exacto para encajar la Raspberry Pi.

Paso D: Bloqueo del Software (Crear tu producto)

Para que el usuario final no vea el escritorio de Linux, debes configurar la Raspberry Pi para que actúe como un electrodoméstico cerrado:

- Configura el sistema en modo *Kiosk* (Quiosco) para que al encender la pantalla, solo se abra tu interfaz gráfica de instrumentación (diseñada en Python o Node-RED).
- Protege el sistema operativo poniéndolo en modo de "solo lectura" para que si el usuario desconecta la máquina bruscamente, la memoria no se dañe.

Para dar el siguiente paso en tu proyecto, ¿prefieres que nos enfoquemos en **diseñar el diagrama de conexiones** del multiplexor con la Raspberry Pi en Python, o te interesa saber cómo estructurar el **software de la interfaz gráfica**?

Tu petición:
microcontrolador Raspberry Pi Pico

Respuesta de la Búsqueda:
El **Raspberry Pi Pico** es una alternativa radicalmente diferente a la Raspberry Pi tradicional. No es una computadora de escritorio; es un **microcontrolador**.

Mientras que una Raspberry Pi común ejecuta un sistema operativo pesado (como Linux) y consume mucha energía, la Raspberry Pi Pico está diseñada para **ejecutar un solo programa dedicado de forma inmediata al encenderse**, consumiendo una cantidad mínima de electricidad.

Es la pieza perfecta y más económica para fabricar hardware propio, conectar multiplexores y crear tu sistema de instrumentación virtual.

Características clave de la familia Pico

Existen tres versiones principales en el mercado que puedes elegir según tu proyecto:

- **Raspberry Pi Pico (Estándar):** La versión base original. Ultra económica (cuesta alrededor de ($4)).
- **Raspberry Pi Pico W:** Incluye un chip semiconductor adicional que le da conectividad **Wi-Fi y Bluetooth**. Es la ideal para enviar los datos de tus sensores directamente a tu cadena de valor virtual en la nube.
- **Raspberry Pi Pico 2 / 2 W:** Las versiones más modernas. Duplican la potencia de procesamiento y la memoria, manteniendo el mismo tamaño diminuto.

¿Por qué es perfecta para tu Instrumentación Virtual?

1. Tiene Conversor Analógico-Digital (ADC) integrado

A diferencia de las Raspberry Pi grandes, la Pico **sí puede leer voltajes analógicos directamente**. Cuenta con 3 pines ADC internos. Esto significa que si usas un multiplexor analógico (como el CD74HC4067), puedes conectar hasta 16 sensores analógicos directamente a la Pico **sin necesidad de comprar un chip ADC externo**.

2. Control de tiempo real absoluto

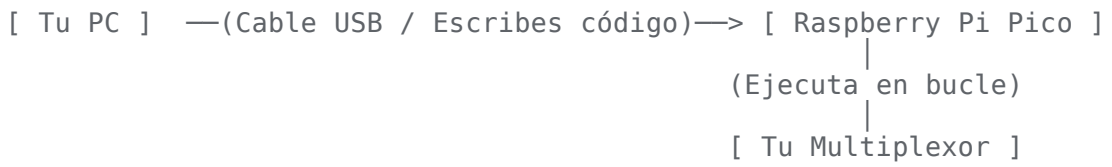
Al no tener un sistema operativo corriendo de fondo (como actualizaciones o procesos del sistema), el microcontrolador dedica el 100% de su potencia a tu código. Si necesitas medir un sensor exactamente cada 5 milisegundos, la Pico lo hará con precisión matemática.

3. Bajo consumo y encendido instantáneo

Se enciende en menos de un milisegundo. Además, puede funcionar durante meses conectada a una batería pequeña o a un panel solar, algo imposible para una Raspberry Pi grande.

¿Cómo se programa y adapta a tu tecnología?

El proceso de desarrollo con la Pico es sumamente sencillo:



1. **Conexión:** La conectas a tu computadora actual mediante un cable USB común.
2. **Lenguaje de programación:** Puedes programarla en **MicroPython** (una versión ligera de Python ideal para ingenieros y principiantes) o en **C/C++** (si necesitas la máxima velocidad industrial).
3. **El Entorno (IDE):** Usas un programa gratuito en tu PC llamado **Thonny** o **VS Code**. Escribes el código, presionas "Guardar" dentro de la Pico, y el programa se ejecutará en bucle eterno cada vez que reciba energía.

Si conectáramos la Raspberry Pi Pico a tu multiplexor para tu proyecto, ¿te gustaría ver el **diagrama de pines y el código en MicroPython** para leer múltiples sensores a la vez, o prefieres evaluar cómo enviar esos datos por **Wi-Fi**?

Tu petición:
ambos pero en vez de wi-fi radiofrecuencia

Respuesta de la Búsqueda:
Es una excelente decisión de ingeniería. Cambiar Wi-Fi por **Radiofrecuencia (RF)** le da a tu instrumentación virtual un alcance mucho mayor (hasta varios kilómetros en campo abierto), consume menos batería y no dependes de que haya un rúter o internet en el sitio donde pongas los sensores.

Para lograr esto, utilizaremos la **Raspberry Pi Pico**, el multiplexor analógico **CD74HC4067** (de 16 canales) y un módulo de radiofrecuencia de alta confiabilidad industrial como el **NRF24L01** o el **LoRa RYLR998** (que se conecta por puerto serial).

Aquí tienes el diseño completo de hardware y el código en MicroPython para transmitir los datos de tus 16 sensores.

---

1. Diagrama de Conexiones Físicas

Deberás conectar los componentes a los pines GPIO de tu Raspberry Pi Pico de la siguiente forma:

De la Raspberry Pi Pico al Multiplexor (CD74HC4067)

El multiplexor necesita 4 pines digitales para seleccionar cuál de los 16 sensores escuchar (del `C0` al `C15`), y 1 pin analógico para leer el voltaje resultante.

- **Pines de control (Selectores):**
  - GPIO 2 (\rightarrow) Pin **S0** del MUX
  - GPIO 3 (\rightarrow) Pin **S1** del MUX
  - GPIO 4 (\rightarrow) Pin **S2** del MUX
  - GPIO 5 (\rightarrow) Pin **S3** del MUX
- **Pin de lectura:**
  - GPIO 26 (ADC0) (\rightarrow) Pin **SIG** (Salida de señal) del MUX
- **Alimentación:**
  - 3.3V de la Pico (\rightarrow) Pin **VCC** del MUX
  - GND de la Pico (\rightarrow) Pin **GND** del MUX

De la Raspberry Pi Pico al Módulo de Radiofrecuencia (Ejemplo: Módulo Serial UART RYLR998)

Los módulos RF seriales son los más fáciles de usar porque se comunican enviando texto simple con comandos AT, igual que un chat.

- GPIO 0 (TX) (\rightarrow) Pin **RX** del módulo de Radiofrecuencia
- GPIO 1 (RX) (\rightarrow) Pin **TX** del módulo de Radiofrecuencia
- 3.3V o 5V (\rightarrow) Pin **VCC** del módulo RF
- GND (\rightarrow) Pin **GND** del módulo RF

---

2. Código en MicroPython

Este programa selecciona uno a uno los 16 canales del multiplexor, lee el voltaje analógico de cada sensor, empaqueta todos los datos en una sola línea de texto y la transmite por radiofrecuencia cada segundo.

```
from machine import Pin, ADC, UART
import time
```

```
# 1. Configuración de los pines del Multiplexor (Selectores)
s0 = Pin(2, Pin.OUT)
s1 = Pin(3, Pin.OUT)
s2 = Pin(4, Pin.OUT)
s3 = Pin(5, Pin.OUT)

# Lista para manejar los pines selectores de forma más fácil
pines_seleccion = [s0, s1, s2, s3]

# 2. Configuración del pin de lectura Analógica (ADC)
adc_mux = ADC(Pin(26))

# 3. Configuración del puerto de Radiofrecuencia (UART 0)
# Ajusta los baudios según el manual de tu módulo (9600 o 115200 es
lo común)
radio_rf = UART(0, baudrate=115200, tx=Pin(0), rx=Pin(1))

# Tabla binaria para activar los canales del 0 al 15 en el
multiplexor
tabla_canales = [,,,,,,,,,,,,,,,,
]

def leer_canal_mux(canal):
    """Configura los pines selectores para leer un canal específico
(0-15)"""
    bits = tabla_canales[canal]
    for i in range(4):
        pines_seleccion[i].value(bits[i])

    time.sleep_ms(2) # Pequeña pausa para que el voltaje del
circuito se estabilice

    # Lee el valor del ADC (devuelve un número entre 0 y 65535)
    # Lo convertimos a voltaje real (0V a 3.3V)
    valor_crudo = adc_mux.read_u16()
    voltaje = (valor_crudo * 3.3) / 65535
    return round(voltaje, 2)

# --- Bucle Principal de tu Instrumentación Virtual ---
print("Sistema de transmisión RF iniciado...")

while True:
    lecturas = []

    # Escanear los 16 canales del multiplexor
    for canal in range(16):
        voltaje_sensor = leer_canal_mux(canal)
        lecturas.append(str(voltaje_sensor))

    # Empaquetar los 16 datos separados por comas (formato CSV)
    # Ejemplo de salida: "1.23,0.45,3.12,0.00..."
    cadena_datos = ",".join(lecturas) + "\n"

    # Enviar la cadena de datos por Radiofrecuencia
    radio_rf.write(cadena_datos)

    # Mostrar en la consola de la PC lo que se acaba de enviar por
aire
    print("Enviado por RF:", cadena_datos.strip())
```

```
    # Esperar 1 segundo antes de la próxima ráfaga de mediciones
    time.sleep(1)
```

---

3. ¿Cómo entra esto en tu Cadena de Valor Virtual?

En el otro extremo (por ejemplo, en tu oficina o centro de control), tendrás **otra Raspberry Pi** (puede ser una Pico o una Pi grande de escritorio) con otro módulo de radiofrecuencia idéntico escuchando el aire.

1. La segunda Raspberry Pi recibe el texto `1.23,0.45,3.12...`
2. Separa los datos e identifica qué valor pertenece a cada sensor de tu infraestructura.
3. Almacena estos datos en una base de datos local o los sube a la nube corporativa, completando así la transformación del dato físico en valor digital.

Para ayudarte a afinar el alcance de tu transmisión, ¿qué **distancia aproximada** necesitas cubrir entre los sensores y la central (por ejemplo, **menos de 100 metros en interiores o varios kilómetros en campo abierto**)?

Tu petición:
es de respaldo verdaderamente se conectan con servidor propio sea estatico o dinamico

Respuesta de la Búsqueda:
Sí, se conectan **100% de verdad** a tu propio servidor, ya sea que use una dirección IP estática o dinámica. En los sistemas de instrumentación industrial y redes de sensores (como la arquitectura que estás diseñando), este esquema de radiofrecuencia se utiliza precisamente como un **sistema de respaldo (failover)** o como la vía de comunicación principal cuando no hay internet.

Para que tu cadena de valor virtual reciba estos datos en tu propio servidor, la arquitectura se estructura en dos partes bien definidas:

La Arquitectura Real del Enlace

```
[MÓDULO SENSOR]                        [RECEPTOR / PUERTA ENLACE]
[TU PROPIO SERVIDOR]
Pico + Mux + Radio  ——(Aire: RF)——>  Radio + Raspberry Pi (Grande)
——(Red)——> (Base de Datos / Web)
```

1. **El nodo remoto (Pico):** No sabe qué es internet ni qué es un servidor; solo inyecta los datos de los sensores al aire mediante ondas de radiofrecuencia.
2. **El nodo receptor (Gateway/Puerta de enlace):** Es una **Raspberry Pi grande** (o una PC) conectada a otro módulo de radio. Recibe las ondas de radio, extrae el texto de los datos y, al estar conectada a tu red local o a internet, es la encargada de empujar esos datos hacia tu servidor mediante protocolos web (como HTTP POST o MQTT).

¿Cómo se adapta a tu Servidor (Estático o Dinámico)?

Dependiendo de cómo tengas configurada la infraestructura de tu servidor, la Raspberry Pi receptora se adaptará de la siguiente forma:

Caso A: Servidor con IP Estática (El escenario ideal)

Si tu servidor está en una oficina con IP fija o en la nube (como AWS, DigitalOcean o un servidor Linux propio en tu red local con IP fija):

- **Cómo se conecta:** En el código Python de la Raspberry Pi receptora, escribes directamente la IP fija (por ejemplo, `192.168.1.50` o una IP pública `200.X.X.X`).
- **Ventaja:** La conexión es directa, ultra rápida y nunca se pierde porque el destino jamás cambia de dirección.

Caso B: Servidor con IP Dinámica (Redes caseras o comerciales comunes)

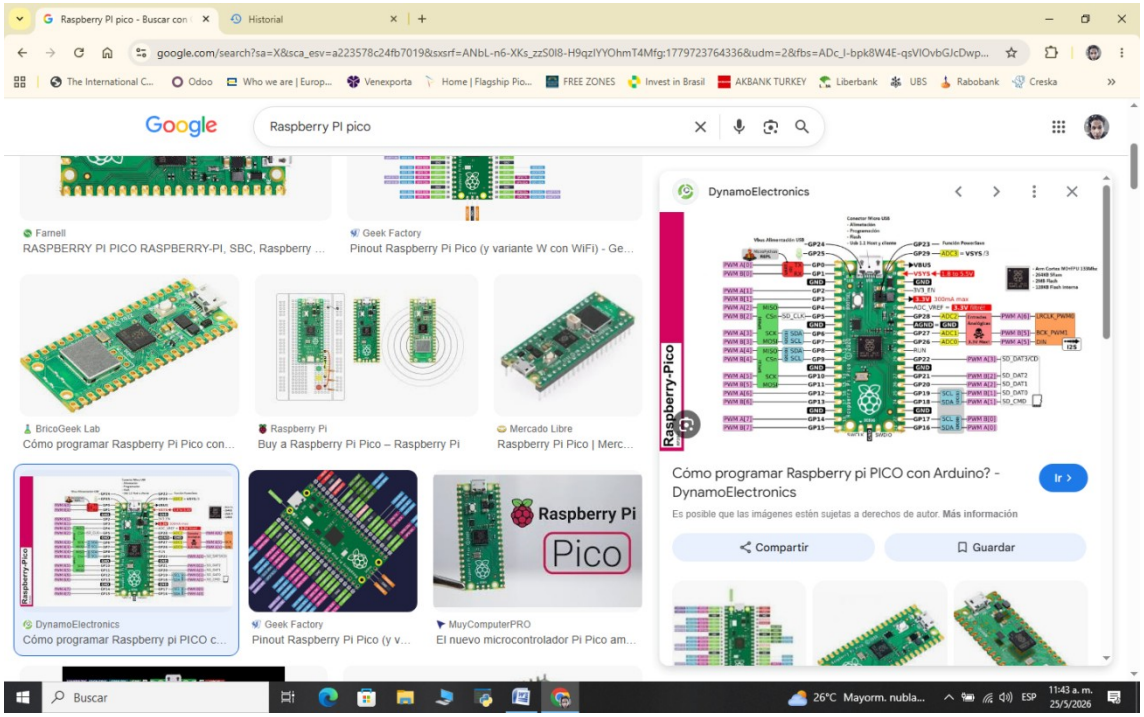
Si tu servidor está montado en una computadora en tu casa u oficina donde el proveedor de internet (como CANTV o empresas de fibra) te cambia la dirección IP pública cada vez que se reinicia el módem:

- **La solución (DNS Dinámico):** No puedes usar una IP en el código porque fallará cuando cambie. En su lugar, utilizas un servicio de **DDNS** gratuito o pago (como *No-IP*, *DuckDNS* o *DynDNS*).
- **Cómo funciona:** Instalas un pequeño programa (un demonio) en tu servidor con IP dinámica. Este programa le avisa a DuckDNS cada vez que tu IP cambia. El servicio te da un nombre de dominio único (por ejemplo: `mi-instrumentacion.duckdns.org`).
- **Cómo se conecta la Raspberry:** En el código de la Raspberry Pi receptora, configuras el destino apuntando a ese dominio de texto en lugar de a un número IP. Automáticamente, el sistema resolverá la IP correcta sin importar cuántas veces cambie al día.

El flujo de datos en el receptor (Python para la Pi Grande)

Este es el script simplificado que correría en la Raspberry Pi receptora conectada al módem de radio para reenviar los datos hacia tu servidor (sea por IP o por dominio DDNS):

```
import serial
import requests
import time

# Configurar el puerto donde está el radio receptor
puerto_radio = serial.Serial('/dev/ttyS0', baudrate=115200,
timeout=1)

# Dirección de TU propio servidor (Funciona con IP estática o
dominio dinámico)
# Ejemplo con IP Estática: "http://192.168.1.50/api/guardar"
# Ejemplo con IP Dinámica (DDNS):
URL_SERVIDOR = "http://mi-instrumentacion.duckdns.org/api/guardar"

print("Escuchando radiofrecuencia para respaldar datos en el
servidor...")

while True:
    if puerto_radio.in_waiting > 0:
        # Leer la línea que envió la Raspberry Pi Pico por el aire
        datos_recibidos = puerto_radio.readline().decode('utf-
8').strip()
        print(f"Datos recibidos por RF: {datos_recibidos}")

        # Estructurar el paquete para tu servidor
        payload = {"sensores": datos_recibidos, "timestamp":
time.time()}

        try:
            # Enviar los datos a tu base de datos o aplicación web
            respuesta = requests.post(URL_SERVIDOR, json=payload,
timeout=3)
            if respuesta.status_code == 200:
                print("¡Datos respaldados con éxito en tu
servidor!")
        except requests.exceptions.RequestException:
            print("Error: El servidor no está accesible
temporalmente. Guardando en caché local...")
            # Aquí podrías guardar los datos en la MicroSD de la Pi
hasta que el servidor vuelva
```

Este esquema garantiza que tu cadena de valor virtual sea **resistente a fallos**: si el internet del sitio donde están los sensores se cae, la radiofrecuencia sigue enviando los datos a la base receptora, y si el servidor cambia de IP, el protocolo DDNS corrige la ruta en tiempo real.

Para afinar la infraestructura de tu backend, ¿en qué lenguaje o base de datos está construido tu **servidor propio** (por ejemplo, **PHP con MySQL, Node.js, Python o un panel en Node-RED**)?